استاد رامتین خسروی درس مهندسی نرم افزار 2 پاییز 1403
اعضای تیم: محمد امانلو(810100084) - محمدعلی قهاری(810100201) - محمدرضا علوی(810100253)

1- چه سازوکاری در اسکرام Insufficient user input را به طور سیستماتیک رفع می‌کند؟

دلیل اصلی موفقیت پروژه‌های IT، مشارکت کاربران نهایی است. پروژه‌هایی که از مشارکت اولیه کاربران برخوردار نیستند، با خطر سوءتفاهم در نیازمندی‌های پروژه مواجه می‌شوند و در برابر پدیده‌ی تغییرات مداوم نیازمندی‌ها آسیب‌پذیرتر هستند. در متن آمده است:

A 1994 Standish Group survey, “Charting the Seas of Information Technology,” found that the primary reason IT projects succeed is because of end-user involvement. Projects without early end-user involvement increase the risk of misunderstood project requirements and are especially vulnerable to time-consuming requirements creep.

متن می گوید که مشارکت اولیه و فعال کاربران نهایی، به کاهش سوءتفاهم‌ها و جلوگیری از تغییرات غیرضروری نیازمندی‌ها کمک می‌کند. در اسکرام، جلسات مستمر مانند جلسات برنامه‌ریزی اسپرینت (Sprint Planning)، اسپرینت‌های کوتاه‌مدت، و جلسات بازبینی (Sprint Review) امکان حضور و مشارکت فعال کاربران نهایی و ذینفعان را فراهم می‌کنند.

product owner نماینده ی اصلی ذی نفعان و کاربران است یکی از مسئولیت های او product backlog است که شامل اولویت ها و نیاز ها و خواسته های کاربران است او با کاربران در ارتباط است و نیاز های آن ها را به تیم توسعه منتقل می کند.

در جلسه sprint review که بین تیم توسعه و ذی نفعان انجام می شود نتایج مورد بررسی قرار می گیرد در این جلسه بازخورد مستقیم از ذی نفعان گرفته می شود و در رفع Insufficient user input تاثیر دارد.

در جلسه ی sprint planning مالک محصول با تیم توسعه کار می کند و نیاز های کاربران را به تیم منتقل می کند و باعث رفع Insufficient user input می شود.

جلسات daily scrum به تیم کمک می کند که به صورت روزانه از پیشرفت و چالش ها مطلع شود و در صورت وجود نواقص آن ها را رفع کند.

اسکرام یک فرآیند تکرارشونده است که بر در آن Inspection and Adaptation وجود دارد و این به تیم این اجازه را می دهد که نواقص از طرف کاربران را پیدا و رفع کند.

این جلسات به تیم‌ها این فرصت را می‌دهد که در هر اسپرینت، بازخوردهای کاربران را دریافت کرده و نیازمندی‌ها را به صورت مداوم با مشارکت آنها بازبینی و به‌روز کنند. در نتیجه، با شفاف‌سازی مستمر نیازمندی‌ها و دریافت بازخوردهای به‌موقع، احتمال سوءتفاهم در نیازمندی‌ها و تغییرات ناخواسته به‌طور چشمگیری کاهش می‌یابد.

2- چگونه روش‌های چابک با Abandoning planning under pressure مقابله می‌کنند؟

روش‌های چابک با تأکید بر برنامه‌ریزی مداوم و بازبینی در طول اسپرینت‌ها، به‌طور موثری با مشکل رها کردن برنامه‌ریزی تحت فشار مقابله می‌کنند. به جای کنار گذاشتن برنامه‌ها هنگام فشار، این روش‌ها انعطاف‌پذیری بیشتری به تیم‌ها می‌دهند و از طریق بازبینی‌های دوره‌ای، تطبیق مستمر با شرایط جدید و تعامل پیوسته با ذینفعان، تیم را از انحراف از مسیر برنامه‌ریزی شده دور می‌کنند. همان طور که در متن آمده است:

Abandoning planning under pressure. Project teams make plans and then routinely abandon them (without replanning) when they run into schedule trouble. Without a coherent plan, projects tend to fall into a chaotic code-and-fix mode, which is probably the least effective development approach for all but the smallest projects.

در متن آمده که تیم‌های پروژه، برنامه‌ها را ایجاد می‌کنند و سپس در زمان مواجهه با مشکلات زمانی، بدون بازبینی آن‌ها را رها می‌کنند. این رویکرد موجب می‌شود که پروژه‌ها به حالت chaotic code-and-fix دچار شوند که معمولاً ناکارآمدترین روش توسعه برای پروژه‌ها است. روش‌های چابک با ایجاد نقاط بازبینی منظم، تیم‌ها را وادار به برنامه‌ریزی و انطباق دوباره می‌کنند تا از این حالت آشفته جلوگیری شود.

در اسکرام تکرار پذیری و چرخه های کوتاه به نام sprint وجود دارد که کار به این چرخه ها تقسیم می شود و در هر چرخه تیم روی یک سری وظایف کوچک و قابل مدیریت تمرکز می کند با این روش تیم ها به جای برنامه ریزی بلند مدت و دقیق برای کل پروژه برنامه ریزی را به بخش های کوچک تر تقسیم می کند و برنامه ریزی در هر چرخه با توجه به شرایط انجام می شود.

در رویکرد چابک تغییرات و تطبیق مداوم بخشی طبیعی از فرآیند است با این کار به جای یک برنامه ی ثابت برنامه در طول پروژه اصلاح می شود. در شرایط فشار به جای کنار گذاشتن برنامه ریزی آن را با واقعیت های جدید تطبیق می دهند.

در روش های چابک برنامه ریزی به صورت مداوم و در فاصله های زمانی مشخص انجام می شود جلسه های sprint planning به تیم اجازه می دهد تا برنامه ی خود را برای یک دوره ی زمانی مشخص کند. این باعث می شود برنامه ریزی تحت فشار زمانی کمتر تحت تاثیر قرار گیرد.

در روش های چابک بازبینی و بازخورد مداوم از ذی نفعان و کاربران یکی از ویژگی های اساسی است در جلسات sprint review بازخورد مستمر از کاربران دریافت می شود و به برنامه ریزی کمک می کند.

در هر sprint تیم یک sprint goal مشخص می کند که باعث می شود تیم روی یک بخش کوچک پروژه متمرکز شود و بتواند برنامه ریزی کند.

روش های پابک به تیم اجازه می دهند که گروه های کوچک تر در تیم به صورت مستقل وظایف خود را مشخص کنند و بین آن ها شفافیت و همکاری وجود داشته باشد که باعث می شود در شرایط فشار هر یک از اعضای تیم وظیفه ی خود را بداند و بتواند برنامه ریزی کند.

3- کدام‌یک از 12 اصل توسعه چابک بر پرهیز از "Wasting time in the "fuzzy front end تأکید دارد؟

اصل سوم:

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Frequent Delivery of Working Software این اصل می گوید به جای برنامه ریزی دقیق و بلند مدت و تلف کردن وقت در مراحل ابتدایی تیم نسخه های کوچک و قابل تحویل را به طور مداوم ارائه دهد.

همچنین در وهله بعدی می‌توان اصل اول را نیز به دلیل توجه به early and continuous delivery بعنوان یکی از اصولی که بر این موضوع تاکید دارد در نظر داشت.

Our highest priority is to satisfy the customer through the early and continuous delivery of valuable software.

این اصل به ما می‌گوید که تحویل سریع و مستمر ارزش به مشتری، یکی از اولویت‌های اصلی توسعه چابک است. این موضوع از اتلاف زمان در مراحل ابتدایی (fuzzy front end) جلوگیری می‌کند، زیرا به جای صرف زمان زیاد در مراحل تایید و برنامه‌ریزی، تیم تمرکز خود را بر روی توسعه مستمر و دریافت بازخوردهای سریع از مشتری قرار می‌دهد. در متن اصلی آمده است که:

Wasting time in the 'fuzzy front end.' This is the time before the project starts, the time normally spent in the approval and budgeting process. It’s easier, cheaper, and less risky to shave a few weeks or months off the fuzzy front end than it is to compress a development schedule by the same amount. But it’s not uncommon for a project to

spend months or years on these preliminaries and then to burst out of the starting gates with an aggressive, often unattainable schedule.

مرحله مبهم ابتدایی به زمانی اطلاق می‌شود که قبل از شروع پروژه، صرف فرآیندهای تایید و بودجه‌بندی می‌شود. پروژه‌ها اغلب در این مرحله زمان زیادی را از دست می‌دهند، و بعد از این مرحله با یک برنامه تهاجمی و غیرقابل‌دسترس شروع به کار می‌کنند. اما در رویکرد چابک، این مشکل با تمرکز بر تحویل زودهنگام ارزش و عدم اتلاف وقت در این مراحل ابتدایی حل می‌شود. این روش باعث می‌شود تیم‌ها به جای انتظار برای تکمیل برنامه‌ریزی‌های طولانی و دقیق، با رویکردی انعطاف‌پذیرتر و با دریافت بازخورد سریع از مشتری، به تدریج محصول را توسعه دهند.

4- چه بخشی از اسکرام مانع از Shortchanging quality assurance to improve development speed می‌شود؟

عبارت Shortchanging quality assurance to improve development speed به معنی کاهش یا حذف فعالیت‌هایی که کیفیت یک محصول را تضمین می کنند مانند تست‌ها، بازبینی‌ها، و برنامه‌ریزی تست برای افزایش سرعت توسعه نرم‌افزار است. در اسکرام بخش های مختلفی وجود دارند که از وقوع این اتفاق جلوگیری می کنند بعنوان مثال:
a. بخش Definition of Done یک تعریف برای زمانی که محصول به شرایط انجام شده می رسد ارائه می شود و این تعریف می تواند شامل استاندارد های مختلف باشد که باعث می شود سرعت فدای کیفیت نشود. مثلا پاس شدن تست‌های مشخصی را هدف قرار دهد.

b. در اسکرام Product Owner مسئول ماکسیمم کردن ارزش محصول است و مطمئن می شود محصول با کیفیت مطلوب تحویل داده می شود و در جلسات مختلف اسکرام بر کیفیت محصول اطمینان پیدا می کند که این به معنی تضمین کیفیت است.

c. در اسکرام بازبینی و بازخورد مداوم وجود دارد که باعث حفظ کیفیت محصول می شود. در Sprint Review ها محصولی که در طول یک اسپرینت تولید شده توسط تیم و ذی نفعان مورد بررسی قرار می گیرد تا کیفیت لازم را داشته باشد. در Sprint Retrospective تیم کار های خود را از نظر کیفیت بررسی کرده و برای اسپرینت بعدی بهتر می کند. اسپرینت های کوتاه و قابل مدیریت باعث می شود تیم به جای تمرکز روی حجم زیادی از کار ها روی بخش های کوچک تمرکز کند و کیفیت بالا برود. مورد مهم در واقع تعهد به کیفیت و بازبینی‌های مکرر در هر اسپرینت است. در اسکرام، تیم‌ها به صورت مکرر کدهای خود را ارزیابی و آزمایش می‌کنند و در انتهای هر اسپرینت، محصول قابل استفاده‌ای را تحویل می‌دهند. این فرآیندهای بازبینی و آزمایش منظم، تضمین می‌کنند که تیم در هر مرحله کیفیت را حفظ کرده و از توسعه سریع بدون توجه به کیفیت جلوگیری شود.

d. در اسکرام کل تیم مسئولیت کیفیت محصول نهایی را بر عهده دارند و باید از کیفیت محصول مطمئن شوند و اعضای تیم یکدیگر را مسئول نگه می دارند تا کیفیت در همه ی مراحل رعایت شود.

e. طبق متن یادداشت مک گانل:

On a rush project, team members often cut corners by eliminating reviews, test planning, and all but the most perfunctory testing. This is a particularly unfortunate decision. Short-cutting a day of QA activity early in the project is likely to add 3 to 10 days of unnecessary activity downstream.

همان طور که مشخص است در متن تأکید شده که کاهش کیفیت در تست‌ها و بازبینی‌ها باعث افزایش هزینه و زمان در مراحل بعدی پروژه می‌شود. به این معنی که هرگونه کاهش فعالیت‌های کیفی در اسپرینت‌های اولیه، باعث افزایش بار کاری و مشکلات در اسپرینت‌های بعدی خواهد شد. مثلا حذف یک روز از فعالیت‌های QA در ابتدای پروژه می‌تواند 3 تا 10 روز فعالیت اضافی و غیرضروری را در مراحل بعدی به همراه داشته باشد.

5- این فهرست از اشتباهات کلاسیک چگونه بند اول بیانیه چابکی Individuals and Interactions over Processes and Tools را موجه می‌کند؟

این اصل می گوید موفقیت پروژه های نرم افزاری به تعاملات انسانی و همکاری تیمی وابسته است یکی از اشتباهات عدم ارتباط کافی با کاربران و ذی نفعان است که باعث می شود نیاز های آن ها به درستی پیاده سازی نشود که این اصل با تاکید به ارتباط مستقیم و مستمر از این اتفاق جلوگیری می کند.

a. Undermined Motivation: روحیه افراد در بلند مدت بیشترین تاثیر را در کارایی و کیفیت دارد و از این رو، اصل اولیه بیانیه چابک، به اهمیت افراد و روحیه آنها در حفظ عملکرد تیم در طولانی مدت اشاره دارد.

1. Study after study has found that motivation probably has a larger impact on productivity and quality than any other factor. Considering that, you would expect a full-fledged motivation program to occupy a position of central importance on every software project. Every organization knows that motivation is important, but only a few organizations do anything about it. Many common management practices are penny-wise and pound-foolish, trading huge losses in motivation and morale for minor methodology improvements or dubious budget savings

b. Uncontrolled Problem Employees: این مورد به اهمیت ارتباط افراد حاضر در تیم اشاره می‌کند. هنگامی ترک یک عضو مخل در تیم در روحیه و همکاری سایر اعضای تیم اثر منفی دارد و در نتیجه باعث کاهش کیفیت عملکرد تیم می‌شود.

At best, failure to deal with problem employees undermines the morale and motivation of the rest of the team. At worst, it increases turnover among the good developers and damages product quality and productivity.

c . یکی از اشتباهات کلاسیک عدم کنترل ویژگی‌های جدید (Feature Creep) است، که در آن بدون ارتباط و تعامل کافی با تیم و کاربران، تغییرات به پروژه تحمیل می‌شود. در نتیجه، نیاز به تعاملات بیشتر میان اعضای تیم و کاربران برای جلوگیری از این مشکل مشخص می‌شود در متن آمده که

Projects without early end-user involvement increase the risk of misunderstood project requirements and are especially vulnerable to time-consuming requirements creep

یعنی پروژه‌هایی که بدون دخالت اولیه کاربر نهایی انجام می‌شوند، خطر عدم درک درست نیازها را افزایش می‌دهند و به ویژه به مشکلات تغییرات مکرر نیازمندی‌ها حساس هستند.

d. علاوه بر این، در اشتباهات کلاسیک دیگری مثل دفترهای شلوغ و پر سر و صدا، اهمیت فضای مناسب برای تعاملات تیمی برجسته شده است. محیط‌های ناکارآمد باعث کاهش بهره‌وری و آسیب به تعاملات افراد می‌شود در متن آمده که:

Workers who occupy noisy, crowded work bays or cubicles tend to perform significantly worse than workers who occupy quiet, private offices.

یعنی در فضای کاری پر سر و صدا، تعاملات افراد و بهره‌وری به شدت کاهش می‌یابد، و این تأکید مجددی است بر اینکه موفقیت تیم‌ها به کیفیت تعاملات و محیط کاری بستگی دارد، نه صرفاً به ابزارها یا فرآیندهای استاندارد شده.

e. مورد بعدی اضافه کردن توسعه‌دهندگان به پروژه‌های دیرکرده است که نشان می‌دهد که تمرکز صرف بر فرآیند، بدون در نظر گرفتن تعاملات افراد، اشتباه است. این امر به وضوح بیان شده:

Adding developers to a late project... new people subtract more productivity from existing staff than they add through their own work.

افزایش افراد بدون توجه به تعاملات و تأثیرات آن بر سایر اعضای تیم، می‌تواند مشکلات را بدتر کند. این موضوع نیز با بند اول بیانیه چابکی هم‌خوانی دارد که می‌گوید تعاملات انسانی باید اولویت بالاتری نسبت به فرآیندها و ابزارها داشته باشد.

6- در مورد Lack of feature-creep control، به نظر می‌رسد منظور مک کانل بیشتر معطوف به Scope Creep است تا Feature Creep. اگر این فرض را بپذیریم، توضیح دهید در فرآیند اسکرام چگونه می‌توانیم با این مشکل برخورد کنیم. به این منظور درباره عوامل این مشکل تحقیق کنید (با ذکر منابع) و برای هر کدام ذکر کنید، این راه‌حل‌ها می‌توانند از انواع زیر باشند:

- **یک یا چند جزء از اسکرام این عامل را رفع می‌کنند**
- **لازم است تیم علاوه بر فرآیند اسکرام، فعالیت‌هایی را برای رفع این عامل به‌کار گیرد**
- **رفع عامل مذکور در محدوده کار تیم تولید نرم‌افزار صورت می‌گیرد**
- **اساساً در توسعه چابک این عامل به‌عنوان مشکل مطرح نمی‌شود**

برای توجیه راه‌حل‌هایی که ارائه کرده‌اید، مختصراً استدلال کنید.

اگر روش‌های چابک را مبنای کار خود داشته باشیم، راه‌حل مک کانل برای حل مشکل را چگونه ارزیابی می‌کنید؟ برای این که حسی از فرآیندهای کنترل تغییر داشته باشید می‌توانید مدیریت درخواست تغییر از فرآیند RUP را مطالعه نمایید.

مشکل Scope Creep به معنی اضافه شدن نیازمندی‌ها و ویژگی‌های جدید به پروژه بدون مدیریت مناسب است، که می‌تواند به افزایش کار، زمان و هزینه‌های پروژه منجر شود. در اسکرام، از چندین مکانیسم برای جلوگیری از این مشکل استفاده می‌شود.

a. یک یا چند جزء از اسکرام این عامل را رفع می‌کنند:

- عدم وضوح نیازمندی‌ها (Unclear Requirements)
یکی از دلایل اصلی Scope Creep، عدم تعریف دقیق نیازمندی‌ها در ابتدا یا تغییر آن‌ها در طول پروژه است.
راه‌حل:
در اسکرام، جلسات Backlog Grooming و Sprint Planning به تیم کمک می‌کنند تا نیازمندی‌ها را به صورت دقیق و کاملاً قابل درک تعریف کند. همچنین Product Owner نقش کلیدی در تعیین اولویت‌ها و شفاف‌سازی نیازها دارد.

- مشارکت ناکافی ذی‌نفعان (Stakeholder Involvement)
زمانی که ذی‌نفعان به درستی در فرآیند توسعه شرکت نکنند، ممکن است نیازهای جدیدی در اواسط پروژه بروز کنند.
راه‌حل:
اسکرام از Sprint Reviews استفاده می‌کند که ذی‌نفعان در انتهای هر اسپرینت نتایج را بازبینی می‌کنند و بازخورد می‌دهند. این جلسات به ذی‌نفعان فرصت می‌دهد تا در فرآیند به‌روزرسانی نیازها مشارکت کنند، بدون اینکه به Scope Creep منجر شود. همچنین لازم است ذی‌نفعان خارج از تیم تولید نرم‌افزار به طور فعال درگیر شوند تا تغییرات نیازمندی‌ها کنترل‌شده و معقول باشند.

- تغییر نیازمندی‌ها (Changing Requirements)
حتی با وجود نیازمندی‌های مشخص، ممکن است در طول پروژه تغییرات اجتناب‌ناپذیر باشند.
راه‌حل:
اسکرام برای مدیریت تغییرات از Backlog Refinement استفاده می‌کند. این فرآیند به تیم اجازه می‌دهد که اولویت‌ها و نیازمندی‌های جدید را در هر زمان با انعطاف‌پذیری بررسی کند، بدون اینکه به تمامی پروژه لطمه وارد شود. Sprint Goals نیز تضمین می‌کند که تغییرات درون اسپرینت تا حد امکان محدود شوند.

- فشار برای افزودن ویژگی‌های جدید (Pressure for New Features)
ذی‌نفعان یا مشتریان ممکن است در طول پروژه، درخواست ویژگی‌های جدید و اضافی کنند که منجر به Scope Creep می‌شود.
راه‌حل:
Product Owner باید به عنوان فیلتر اصلی برای درخواست‌های جدید عمل کند و اطمینان حاصل کند که ویژگی‌های جدید فقط در صورت تأیید و در اولویت مناسب به Backlog اضافه شوند. همچنین باید به تیم کمک کند تا از اضافه‌شدن ویژگی‌های غیرضروری جلوگیری کند.
علاوه بر این، باید در سطح مدیریتی یا با ذی‌نفعان جلسه‌های منظم برگزار شود تا از فشارهای غیرمنطقی کاسته شود و تیم متمرکز بماند.

- عدم کنترل تغییرات (Lack of Change Control)
زمانی که فرایند کنترل تغییرات مناسب وجود نداشته باشد، اضافه‌شدن ویژگی‌های جدید بدون بررسی دقیق رخ می‌دهد.
راه‌حل:
در روش‌های اسکرام و چابک، از Time-Boxing استفاده می‌شود. اسپرینت‌ها دوره‌های زمانی محدود و

ثابتی هستند که در طول آن تغییرات محدود می‌شوند. درخواست‌های تغییرات بزرگ به اسپرینت‌های آینده منتقل می‌شوند تا تیم تمرکز خود را حفظ کند.

b. علاوه بر فرآیند اسکرام، فعالیت‌های بیشتری لازم است:

- برقراری ارتباط موثر: تیم توسعه باید با ذینفعان و مشتری‌ها ارتباط منظم و موثری داشته باشد تا نیازمندی‌های واقعی پروژه را به‌خوبی درک کرده و از درخواست‌های غیر ضروری جلوگیری شود.

- تعیین محدوده دقیق پروژه: باید محدوده دقیق پروژه از ابتدا مشخص شود و هرگونه تغییر به صورت رسمی و با توافق همه ذینفعان وارد پروژه گردد.

c. اساساً در توسعه چابک این عامل به عنوان مشکل مطرح نمی‌شود:

- انعطاف‌پذیری چابک: در چابک و اسکرام، تغییرات مورد انتظار است و فرآیندها به گونه‌ای طراحی شده‌اند که تیم‌ها بتوانند به صورت مداوم با شرایط جدید تطبیق پیدا کنند. بنابراین، اگر نیازمندی‌های جدید با مدیریت مناسب در زمان‌بندی‌های بعدی وارد شوند، Scope Creep به عنوان مشکل نخواهد بود.

Feature and scope creep are rampant. Projects suffer from requirements growth because teams implement the easy requirements first and leave the hard requirements until later. That leaves most projects behind schedule even though they are “almost done.” Scope creep also occurs when requirements are poorly defined and new features are added as the system develops. More formally defined change control processes could help to curb scope creep

طبق توضیحات داریم که Feature Creep و Scope Creep اغلب زمانی رخ می‌دهند که نیازمندی‌ها به درستی تعریف نشده باشند و ویژگی‌های جدید در طول توسعه به سیستم اضافه شوند. در اسکرام، با استفاده از روش هایی که بالاتر توضیح داده شد جلوی این مشکل گرفته می شود.

حال اگر روش‌های چابک را مبنای کار قرار دهیم، راه‌حل مک‌کانل که شامل به‌کارگیری فرآیندهای رسمی‌تر مانند کمیته کنترل تغییرات (Change Control Board یا CCB) است، با اصول چابکی در تضاد قرار می‌گیرد. رویکرد Agile معمولاً به دلیل انعطاف‌پذیری و سرعت، ساختارهای کنترلی پیچیده را کاهش داده و بر بازخورد مستمر و بهبود تدریجی تمرکز دارد. در واقع، ایجاد یک کمیته تغییر ممکن است باعث کند شدن فرآیند و کاهش سرعت تصمیم‌گیری در تیم شود، که با ارزش‌های چابک همخوانی ندارد.

راه‌حل مک‌کانل را می‌توان به صورت محدود و با اصلاحات زیر به کار برد:

ثبت و مستندسازی تغییرات در Backlog: به جای تشکیل کمیته رسمی، تغییرات به‌طور مستقیم در Product Backlog ثبت شوند و Product Owner در مورد اولویت‌بندی و تأیید آنها تصمیم‌گیری کند.

بازبینی مستمر تغییرات در جلسات اسپرینت Review: این جلسه‌ها می‌توانند جایگزین جلسات CCB شده و به تصمیم‌گیری سریع‌تر و چابک‌تر درباره تغییرات کمک کنند.

7- در مورد Shortchanging upstream activities این طور ذکر شده که تیم‌ها تحت فشار زمانی فعالیت‌هایی مثل تحلیل نیازمندی‌ها، طراحی و معماری را حذف می‌کنند. سوالی که در این باره پیش می‌آید این است که روش‌های چابک با توجه به تاکید بر تحویل مکرر و زودهنگام چه میزان روی طراحی توان صرف می‌کنند. به این منظور مقاله "Is Design Dead" را بخوانید و بعد درباره این که آیا ممکن است اشتباه مورد نظر مک‌کانل در روش‌های چابک نیز اتفاق بیفتد بحث کنید.

Shortchanging upstream activities به این معنی است که تیم‌ها برای جبران فشار زمانی، فعالیت‌هایی مثل تحلیل نیازها، طراحی و معماری را حذف می‌کنند. این اقدامات در روش‌های سنتی ممکن است منجر به خطاهای بیشتر و افزایش هزینه‌ها شود. در مقابل، روش‌های چابک با تحویل مکرر و فازبندی فعالیت‌ها، به کاهش این خطاها کمک می‌کنند.

با این حال، حذف طراحی کامل و توجه ناکافی به معماری حتی در روش‌های چابک نیز می‌تواند به مشکلاتی در آینده منجر شود. مک‌کانل اشاره می‌کند که تیم‌هایی که روی فعالیت‌های اولیه طراحی صرفه‌جویی می‌کنند، ممکن است با هزینه‌های بسیار بالاتری در مراحل بعدی مواجه شوند. در متن آمده:

Project teams that are in a hurry try to cut nonessential activities, and because requirements analysis, architecture, and design don’t directly produce code, they are easy targets for the schedule ax. On one disaster project that I took over, I asked to see the design. The team leader told me, “We didn’t have time to do a design.” Also known as “jumping into coding,” the results of this classic mistake are all too predictable.

مک کانل می گوید که یکی از اشتباهات رایج در پروژه های نرم افزاری نادیده گرفتن فعالیت های مهمی مانند تحلیل نیازمندی ها و طراحی و معماری است که به دلیل فشار زمانی اتفاق می افتد. در روش های چابک دلایلی وجود دارد که ممکن است این اشتباهات رخ دهد مانند این که در روش های چابک تاکید بیش از حد روی سرعت تحویل وجود دارد که باعث می شود تیم ها روی سرعت تحویل تمرکز کنند و

برخی از فعالیت ها را انجام ندهند. همچنین در روش های چابک تیم ها ممکن است تحت فشار ذینفعان و مشتریان قرار بگیرند که باعث می شود برخی فعالیت ها را انجام ندهند و روی سرعت تحویل وقت بگذارند.

از طرفی در روش های چابک دلایلی وجود دارد که می تواند باعث شود این اتفاق نیفتد. در اسکرام طراحی به صورت تدریجی انجام می شود و در هر چرخه بهتر می شود این باعث می شود به جای این که از همان ابتدا کامل انجام شود به صورت تدریجی و با تغییرات پروژه انجام شود. این جوری این فعالیت ها نادیده گرفته نمی شود. روش های چابک از ابزار هایی مانند تست های خودکار و ... استفاده می کنند این روش ها باعث می شود تیم با وجود فشار زمانی کیفیت طراحی را حفظ کند. روش های چابک روی فیدبک گرفتن مداوم از مشتریان و کاربران تاکید دارند این باعث می شود که تیم به مرور زمان طراحی را بهبود دهد و از اضافه شدن ویژگی های بدون کاربرد و پایین آمدن کیفیت جلوگیری کند.

در روش های چابک طراحی به عنوان بخش مهمی از فرآیند توسعه است و به صورت تدریجی انجام می شود این به تیم اجازه می دهد با تغییرات پروژه سازگار شوند و کیفیت طراحی را حفظ کنند و فشار زمانی باعث نشود که بعضی از فعالیت ها نادیده گرفته شود.

اما در مقاله Is Design Dead درباره چابک به شرح زیر انتقاداتی رخ داده:

Agile methodologies emphasize adaptive planning and evolutionary development, often under the misconception that design is not needed. However, design in agile is iterative, focusing on what’s essential at each phase, but failing to design upfront can lead to inefficiencies later.

در واقع در مقاله Is Design Dead، نویسنده به این مسئله می‌پردازد که در متدولوژی‌های چابک، طراحی به صورت تکراری و بر اساس نیازهای هر فاز انجام می‌شود. اما این تفکر که طراحی کلی لازم نیست، می‌تواند در مراحل بعدی به ناکارآمدی‌هایی منجر شود.

و این باعث میشود که روش های چابک همچنان در معرض "Short changing upstream activities" قرار دارند. که برخی دلایل آن به شرح زیر است:
نادیده‌گرفتن طراحی بلندمدت: طراحی تکاملی اگرچه مزیت‌هایی مانند انعطاف‌پذیری دارد، اما ممکن است باعث شود که تیم‌ها به جای طراحی بلندمدت و پایدار، تنها به نیازهای فوری و کوتاه‌مدت توجه کنند. این موضوع می‌تواند به بروز مشکلات ساختاری در مراحل بعدی منجر شود.

پیچیدگی پروژه‌های بزرگ‌تر: در پروژه‌های بزرگ‌تر و پیچیده‌تر، ممکن است طراحی تکاملی به اندازه کافی قوی نباشد و عدم توجه به طراحی و معماری در ابتدای پروژه باعث بروز مشکلات جدی در آینده شود.

8- چه نمونه‌هایی از Silver-bullet syndrome سراغ دارید که در سال‌های اخیر مطرح و تبلیغ شده باشند؟

Silver-bullet syndrome اشاره به این دارد که برخی افراد یا سازمان‌ها به دنبال راه‌حل‌های سریع و آسان برای مشکلات پیچیده هستند، در حالی که چنین راه‌حل‌هایی معمولاً غیرواقعی و غیرقابل اعتمادند. در سال‌های اخیر، ابزارها و فناوری‌های مختلفی به عنوان راه‌حل نهایی یا گلوله نقره‌ای معرفی شده‌اند که ممکن است تبلیغ‌شده ولی عملاً مؤثر نباشند.

As technology evolves, new tools and methods are often introduced as magic solutions that will solve all problems. Examples from recent years include blockchain, artificial intelligence, and no-code platforms, which have been heavily promoted but often fall short of their promises when used inappropriately.

در واقع مک کانل اشاره می‌کند که تکنولوژی‌های جدید مانند بلاکچین، هوش مصنوعی، و پلتفرم‌های بدون نیاز به کدنویسی به عنوان «راه‌حل‌های جادویی» معرفی شده‌اند. اما در بسیاری از موارد، وقتی به طور نادرست استفاده می‌شوند، نتایجی که انتظار می‌رود را ارائه نمی‌دهند. در واقع هیچ پیشرفت واحدی در حوزه تکنولوژی یا مدیریت وجود ندارد که به تنهایی بتواند پیشرفت بزرگی را در مدت کوتاه ایجاد کند. به جای آن، بهبودها از تجمع پیشرفت‌های کوچک به دست می‌آیند. حال هر یک از مواردی که در متن اصلی اشاره شده توضیح داده می‌شود:

Microservices Architecture

- توضیح: معماری میکروسرویس‌ها به عنوان جایگزینی برای معماری‌های مونولیتیک معرفی شد که می‌تواند مقیاس‌پذیری، توسعه‌پذیری و مدیریت مستقل سرویس‌ها را بهبود بخشد. این تکنولوژی به‌طور گسترده‌ای به‌عنوان یک راه‌حل کامل و نهایی برای تمام مشکلات مربوط به معماری نرم‌افزار تبلیغ شد.
- واقعیت: هرچند میکروسرویس‌ها برای بسیاری از پروژه‌ها بسیار مفید هستند، اما برای همه پروژه‌ها مناسب نیستند. مشکلاتی مانند پیچیدگی مدیریت ارتباطات بین سرویس‌ها، خطاهای

ناشی از شبکه، هزینه بالای تست و استقرار و نیاز به ابزارهای تخصصی، نشان دادند که میکروسرویس‌ها برای همه راه‌حل کامل نیستند.

NoSQL Databases

- توضیح: پایگاه‌داده‌های NoSQL به‌عنوان جایگزینی برای پایگاه‌های داده رابطه‌ای مطرح شدند که می‌توانند با مقیاس‌پذیری بیشتر و انعطاف‌پذیری بالاتر به مشکلات پایگاه‌های داده سنتی پاسخ دهند.
- واقعیت: با اینکه NoSQL مزایای خاصی مثل ذخیره‌سازی داده‌های بزرگ بدون ساختار دارد، اما عدم پشتیبانی از تراکنش‌های پیچیده، مدل‌های ضعیف برای یکپارچگی داده‌ها و نیاز به دانش خاص برای مدیریت و پیکربندی آن‌ها، این واقعیت را نشان داد که NoSQL نیز در همه شرایط بهترین انتخاب نیست.

Blockchain

- توضیح: بلاکچین به عنوان یک فناوری انقلابی معرفی شد که می‌تواند مشکلات مربوط به اعتماد و امنیت را در بسیاری از صنایع از بین ببرد. این فناوری در ابتدا در بیت‌کوین و ارزهای دیجیتال مورد استفاده قرار گرفت، اما سپس به عنوان راه‌حلی برای کاربردهای مختلفی مانند مدیریت زنجیره تأمین، قراردادهای هوشمند و حتی رأی‌گیری الکترونیکی تبلیغ شد.
- واقعیت: اگرچه بلاکچین دارای پتانسیل‌هایی است، اما محدودیت‌هایی مانند هزینه‌های بالا، سرعت کم تراکنش‌ها، مصرف بالای انرژی و پیچیدگی پیاده‌سازی در بسیاری از پروژه‌ها نشان داد که این فناوری نمی‌تواند به‌عنوان راه‌حلی کامل برای همه مشکلات محسوب شود.

ML (Machine Learning) و AI (Artificial Intelligence)

- توضیح: هوش مصنوعی و یادگیری ماشینی به‌عنوان فناوری‌هایی معرفی شدند که می‌توانند بسیاری از چالش‌ها را به‌صورت خودکار و هوشمند حل کنند. این فناوری‌ها به‌عنوان راه‌حل‌هایی برای بهبود تجربه کاربر، تصمیم‌گیری‌های بهتر و خودکارسازی فرآیندهای پیچیده تبلیغ شدند.
- واقعیت: در حالی که ML و AI توانایی‌های فوق‌العاده‌ای دارند، پیاده‌سازی و نگهداری آن‌ها چالش‌های بزرگی دارد. نیاز به داده‌های عظیم، پردازش‌های پیچیده، و مشکلات اخلاقی و حریم خصوصی، از جمله مواردی هستند که نشان می‌دهند این تکنولوژی‌ها به‌تنهایی برای حل همه مشکلات مناسب نیستند.

Low-Code/No-Code Platforms

- توضیح: پلتفرم‌های Low-Code و No-Code به‌عنوان ابزارهایی برای ساده‌سازی توسعه نرم‌افزار معرفی شدند که به کاربران غیر فنی اجازه می‌دهند بدون نیاز به کدنویسی پیچیده، برنامه‌های کاربردی بسازند.
- واقعیت: هرچند این پلتفرم‌ها برای ساخت اپلیکیشن‌های ساده و کاربردهای سریع مفید هستند، اما محدودیت‌های آن‌ها در پیاده‌سازی پروژه‌های پیچیده و نیاز به برنامه‌نویسان حرفه‌ای برای توسعه ویژگی‌های پیشرفته نشان داد که این پلتفرم‌ها نمی‌توانند جایگزین کامل توسعه سنتی باشند.